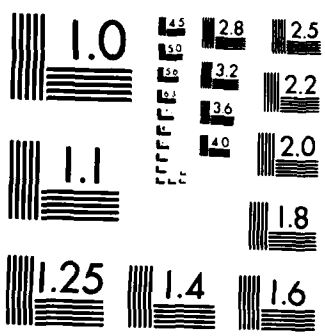
AD-A153 760 DISTRIBUTED DATABASE DESIGN TOOL(U) NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA T BENTSON MAR 85 NOSC/TD-783 1/1

UNCLASSIFIED F/G 9/2 NL



TD 783

Technical Document 783
March 1985

DISTRIBUTED DATABASE DESIGN TOOL

Thomas Bentson

Naval Ocean Systems Center
San Diego, California 92152-5000

NAVAL OCEAN SYSTEMS CENTER SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

F. M. PESTORIUS, CAPT, USN
Commander

R.M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work, sponsored by the Naval Electronic Systems Command (NAVELEX 614), covers the period from May to December 1984. The work addresses the test and evaluation of a class of database management systems (DBMS) algorithms. These algorithms, called "global schemas" and "local schemas," are unique to distributed DBMSs. They govern the location, correlation, storage, retrieval, and deletion of multiple data copies. A distributed DBMS relies on the service of a communications system to transmit both control information (e.g., "read" or "write") and data. The test and evaluation methodology allows the distributed DBMS to be modeled and tested together with the underlying communications system in order to allow these systems to be tuned to each other.

Maniel Vineberg, of the System Design and Architecture Branch, served as technical reviewer on the project.

Released by
Howard Wong, Head
Tactical Systems Division

Under authority of
Harold Smith, Head
Communications Department

SM

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC TD 783	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center 6b OFFICE SYMBOL (if applicable)	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	7b ADDRESS (City, State and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Electronic Systems Command 8b OFFICE SYMBOL (if applicable) NAVELEX 614	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State and ZIP Code) Washington, DC 20363	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 62762N; PROJECT NO: F62582; TASK NO: XF62582110; WORK UNIT NO: 811-EE71

11 TITLE (include Security Classification)
DISTRIBUTED DATABASE DESIGN TOOL

12 PERSONAL AUTHOR(S)
Thomas Bentson

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM May 1984 TO Dec 1984	March 1985	11

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Distributed, Database management, Communications, Schema, Global schema, Local schema

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This document discusses a simulation design tool for distributed database management systems. The ability to model database transaction processing has been added to an existing communications software testbed. The information architecture used is an extension of the ANSI/SPARC standard three-schema DBMS architecture. This extended architecture was developed by the Honeywell Computer Science Center. This document discusses the standard DBMS architecture, the extended distributed DBMS architecture, the logical transaction model of database operations, and the method of simulating this logical model. Measures of data management algorithm effectiveness are also discussed.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	Agency Accession
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT [ ] DTIC USERS		DN088671
22a NAME OF RESPONSIBLE INDIVIDUAL Thomas Bentson	22b TELEPHONE (include Area Code)	22c OFFICE SYMBOL

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS

INTRODUCTION

In the next several years military system designers must develop highly reliable distributed database management systems. A distributed database management system (DBMS) must ensure that critical data are available, accurate, and current. To build an effective DBMS, designers must consider the communications mechanism between remote locations, as well as the inherent relationship between data availability and consistency. For this reason system designers need a tool that supports the test and evaluation of data management and concurrency control algorithms as well as the underlying communications network.

A Network Architecture Simulated Text and Evaluation Environment (NASTEE) has been developed at the Naval Ocean Systems Center (Ref. 1). NASTEE is used to test and evaluate network-based communication protocols. It has been extended to support the modeling and testing of distributed DBMS functions as users of the communications network. The DBMS functions are controlled by user-supplied data managment algorithms. The data management algorithms allocate data copies to sites (nodes); choose execution sites; order, detect, and respond to communications partitions; test for data inconsistency; and resolve inconsistencies when they arise. The testbed generates database transactions at either constant or variable rates at user-specified nodes. In response to transactions, the data management algorithms generate messages to carry information and database operations to remote nodes. These transactions generate some network messages. At remote nodes, received messages use the message data field and type to initiate action on data records.

This report discusses extensions made to a centralized DBMS architecture in order to define a distributed DBMS architecture. The logical view of transaction processing in the distributed DBMS environment is presented with respect to this architecture. The simulation of this logical model by using the NASTEE communications testbed is then described. Some measures of effectiveness for data management algorithms are also discussed.

ANSI/SPARC STANDARD DBMS ARCHITECTURE

A standard architecture for a centralized DBMS was proposed by the ANSI/SPARC Study Group on Data Base Management Systems (Ref. 2). This architecture (shown in Fig. 1) is composed of three schemas, or levels: an external level, a conceptual level, and an internal level. Date (Ref. 3) defines these three levels as follows:

The external level represents an individual user view. There may be many external schemas for any one DBMS. Each of these external views will represent some more or less abstract portion of the database. Each user probably will have interest in or access to only a portion of the total database. The external user view represents the contents of the database to that user.

The conceptual level represents the community view of the database. This is an abstract representation of the data that describes what objects are stored in the total database and how those objects exist in relation to one another. This level represents the entire content of the database. There

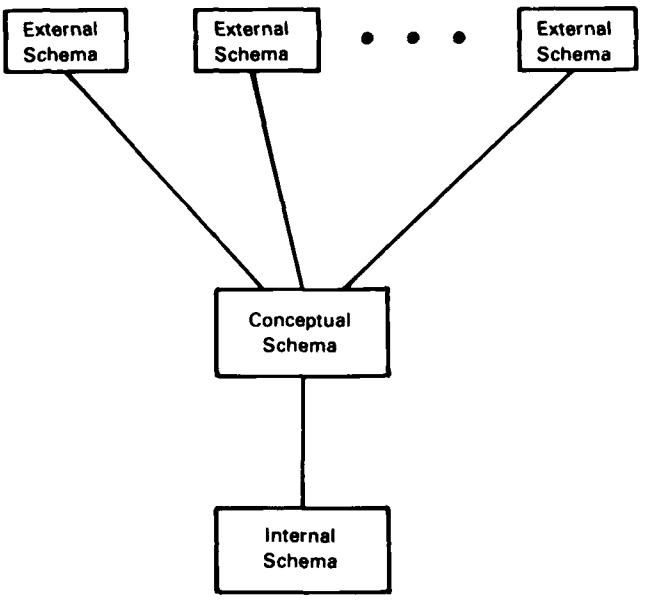


Figure 1. Standard three-schema architecture.

will be only one conceptual schema for each DBMS. A record at the conceptual level may be very different from an external record or an internal record.

The internal level represents how the data are actually stored in and received from the storage devices. This inludes how records are addressed and how storage fields are defined, activities which may be highly dependent on the physical resources available to the system.

The advantage of this architecture is that change may be made at any level without having to redesign the other two. Any level is insulated from the changes at other levels by the interface between levels. This makes updating the system and supporting new user applications less disruptive to the current users.

AN EXTENDED ARCHITECTURE FOR DISTRBUTED DBMS

Honeywell has extended the standard three-schema architecture for a centralized DBMS to a five-schema architecture for a distributed DBMS (Ref. 4,5). The extensions include two additional schemas to perform functions unique to distributed processing. These two schemas are referred to as the global schema and the local representational schema. This extended architecture is shown in Fig. 2. Before describing these two additional schemas it will be helpful to discuss some of the problems unique to distributed processing.

The problem areas in distributed DBMS can be broadly classified as consistency and availability. The problem of consistency arises when there are multiple copies of a file in a database. A database is consistent if each

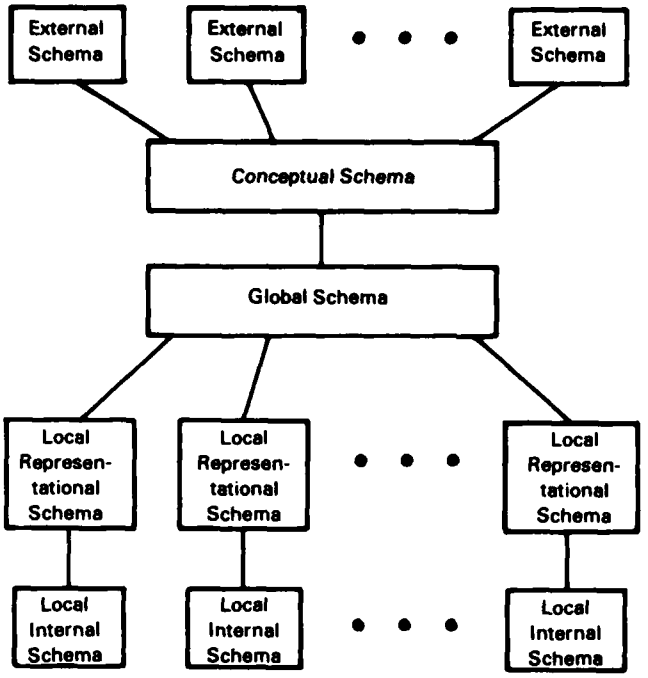


Figure 2. Honeywell's five-schema architecture.

physical copy of a record agrees with other physical copies. In order to maintain consistency, a transaction performed on one physical record must be performed on other physical copies of that record. This is usually accomplished by denying access to a record until all copies of that record have been updated. The second problem, availability, occurs if access to a record is denied or delayed while the communications system access is a remote location. Ensuring consistency and maintaining availability can then be seen as conflicting goals.

The database management needs of the Navy are unique. When data are needed to support critical command and control operations in a hostile environment, availability is just as critical as consistency. Availability becomes a real problem in a hostile environment when communications become partitioned or are unreliable.

The function of the global schema to ensure both availability and consistency implies that the global schema enforces the decisions that make trade-offs between availability and consistency. Other functions of a global schema include selection of operation and data storage sites, coordination of transaction execution, detection of and recovery from data inconsistencies, and implementation of crash recovery algorithms. The global schema is critical to the performance of a distributed DBMS. Central to the performance of the global schema is the communications network which links remote data storage sites and transaction request sites.

The local representational schema at each node is a partial copy of the global schema that describes data stored at the node. In addition, a local representational schema performs transactions, or portions of transactions, in a query language used by the global schema to define the operations scheduled at this node.

A LOGICAL VIEW OF TRANSACTION PROCESSING

Operations on a database are viewed as a set of transactions. Each transaction is processed by two logical entities, an application processor, and a data processor (Ref. 5) (see Fig. 3).

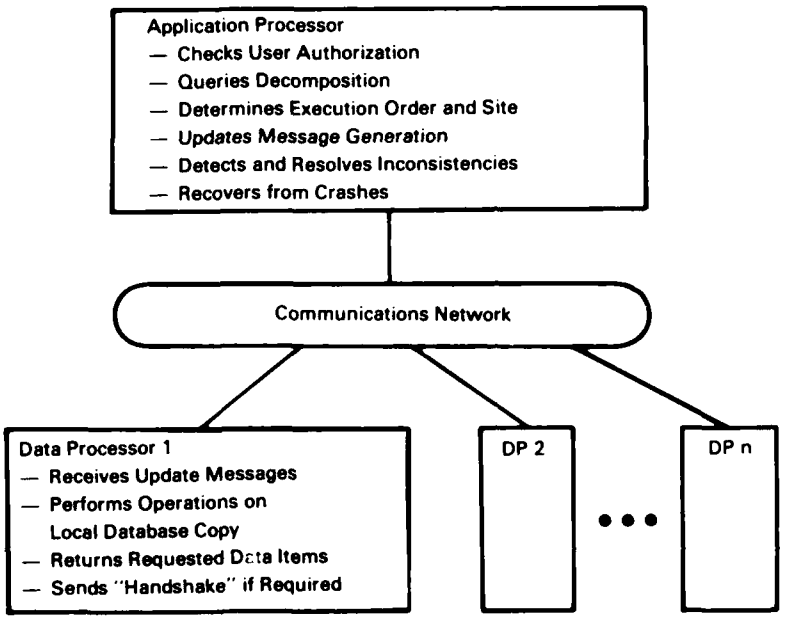


Figure 3. Logical transaction model.

The function of an application processor is to screen applications for modifications to, or requests for information from, a database. In terms of the five-schema architecture, the application processor implements both the conceptual and global schemas. The conceptual schema ensures that the user requesting a transaction is authorized to perform that operation and that the operation is valid. The global schema directs the modifications or inquiries to the appropriate copies in the database. Having identified the database copies, the application processor transmits inquiries and modifications in the appropriate scheduled execution order.

When a data processor receives an inquiry or modification message, it either updates or searches its copy of the database. It then sends any required votes for concurrency control, acknowledgements, or requests for information over the communications system. The destination for these responses is supplied by the global schema and transmitted as part of the operation request message. In terms of the five-schema architecture, the data processor performs some global schema functions and all local representational and local internal schemas functions.

In general, the application processor and the data processor communicate via the communications network. The final schema to be accounted for is the

external schema. This is represented by the nature and format of transaction requests made to the application processor.

MODELING THE LOGICAL VIEW OF TRANSACTION PROCESSING

The logical transaction model is implemented as described below. The purpose of extending NASTEE is to create an environment in which user-supplied global schemas can be tested and evaluated, rather than to accurately model all functions of a distributed DBMS. Thus the supporting environment for the global schema is not designed to use preferred schemas. In fact the nature of schemas at other than the global level is not of particular interest. It is assumed, for simplicity, that the data of interest are track data. A track refers to the stored information about an object in three-dimensional space; this object may be represented by one or more track files at multiple nodes. This simplifying assumption allows us to manipulate data with a single record format and a limited number of data fields. Thus we can operate on data without defining a generic data type or building complex relational operators. The following discussion assumes some knowledge of the SIMSCRIPT II.5 programing language (Ref. 6).

Data elements (tracks) are defined as SIMSCRIPT entities. Each track has the following attributes: x, y, and z position coordinates; a creation time; a most-recent-update time; a pointer to the node that created the track; a track history; a track version number; and a unique global identifier. These attributes allow transactions performed on each data element to be treated and checked for consistency.

Each node in the distributed network owns a set of tracks; this set may be empty. More than one physical track copy may exist for each object. For a given track physical copies may differ (in age or accuracy) from other physical track copies. Multiple copies of a track are still referred to as a single instance of a track. Each track belongs to at least one set of tracks. Thus at each node there is a local database that gives the user the flexibility required to support partial or total redundancy in data storage.

The testbed updates and queries the database. These operations are referred to as transactions. A SIMSCRIPT process to generate transactions at user-specified rates is activated for each transacton type at each node. These transaction generators put a processing load on the distributed DBMS. A transaction type represents an operation that can be performed on the database. The manipulative power of the DBMS can be extended by introducing new transaction types. Initially the types of transactions are restricted to reads and writes. This should be sufficient to model simple database transactions. Transactions of each type are generated at rates that may be constant or that vary dynamically with time; this is a user option. The option of a table-driven transaction generator is also available. The transaction generator activates a transaction as a SIMSCRIPT process and assigns values to the attributes of the transaction. The attributes of a transaction represent the data fields needed to complete a database operation. If appropriate, the transaction generator computes the time of the next transactions.

After a transaction process is activated, it is filed in the transaction queue and suspended. A transaction queue server serves the transaction queue

on a first-in-first-out basis. When reactivated by the transaction queue server, the transaction requests the application processor resident at the node. The application processor is defined as a SIMSCRIPT resource, with one copy of the resource at each node. When access to the resource is granted, the user-supplied global schema processes the transaction. In processing the transaction the global schema interprets the transaction attributes and uses its knowledge of data locations to generate a sequence of data base operations, operation sites, and operation times. These operations are sent as messages to the operation sites. The resource is then released and the transaction queue server reactivates the next transaction in the transaction queue. The attributes of a message are used to carry data and operation type to the destination node. The functional abilities described so far model the logical application processor.

Messages put themselves on to the communications network. Communications protocols are called and the message is transmitted to the destination node. In general, not all messages on the communications network are database operations. The DBMS may be only one of many users of the network. At the receiving node the message is decoded; if it is a database operation, the data processor is activated. This allows flexibility in specifying the communications network used by the DBMS. Support of the distributed DBMS architecture by a communications network is shown in Figure 4.

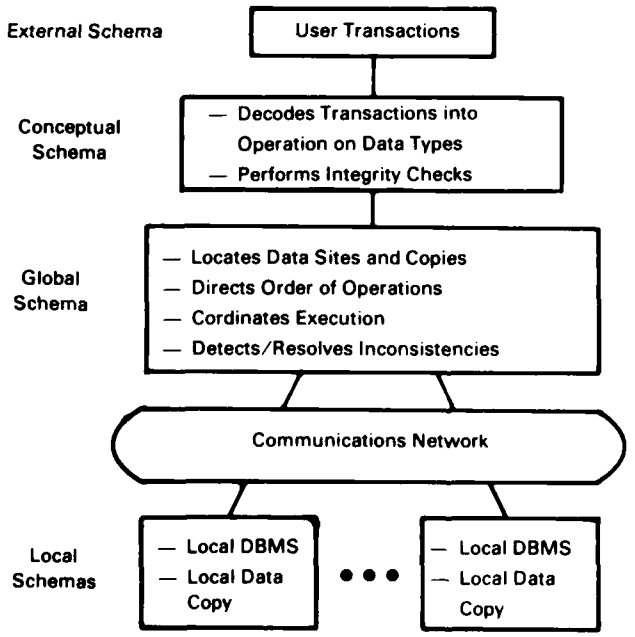


Figure 4. Network support of five-schema architecture.

When a message is received at the destination node a SIMSCRIPT process modeling the data processor (DP) is activated. This DP files itself in a queue. Once it is determined that no existing DP is in conflict with the new database operation, the DP process is reactivated and a resource for

processing is requested. When resource access is granted, the DP calls the user-supplied schema routine in levels. After each of these levels the resource is released, and the process is suspended. The process is reactivated by new incoming messages that are directed at the suspended DP. When reactivated, the DP requests the processing resources again and, when access is granted, calls the next level of schema processing. These levels of processing represent DP functions that require input from remote nodes. When all functions that require remote input are completed, a flag is set so that any further messages intended for the DP are ignored. Once remote nodes have communicated enough information for the transaction to continue, the DP calls an update routine to perform an operation on the local database.

If the operation is a write, the update module verifies that the track is at this node. If the track does not already exist at this node, the DP creates a track and track history file for the track identifier. If the track exists, the current track information is stored as a track history in the track history file, and the new information is written into the track's data field (attributes). This represents the total transaction initiated by a write.

If the operation is a read, the update module writes a short message to the output file indicating that the read was performed and showing the values that were read.

At several points in the transaction and data processing phases, data are collected for use in computing measures of effectiveness for the schema. These measures are discussed in the following section.

GLOBAL SCHEMA MEASURES OF EFFECTIVENESS

To determine measures of effectiveness, it is necessary to define an effective schema. It will help, first, to review the functions of a global schema. The global schema determines execution and storage sites and execution order, generates messages in response to queries, and detects and resolves any inconsistencies. Such inconsistencies might be the result of a communications partition. A communications partition occurs when no copy of a data item is available because the communications system cannot access any (or enough) of the nodes where the data item is stored. A node may not be accessible because of heavy message traffic (long queue delays) or equipment damage. The global schema also implements crash recovery algorithms.

Presently the extended version of NASTEE collects the following measures of effectivness by node and by type: (1) transactions tried, (2) transactions completed, (3) transaction delay, (4) transaction time in queue, (5) DP time in queue, and (6) DP processing time. Computed from collected measures are the following: (7) the ratio of transactions completed to transactions tried and (8) the ratio of messages sent to transactions tried. A measure of the partition detect time should be added.

The first measure collected by NASTEE, transactions tried, indicates the load on the DMBS. The numbers and types of transactions requiring processing at each node are recorded. Although transactions are generated at user-defined rates, some transactions generated may not be valid transactions

(parameters out of bounds, incorrect track identification). These will be detected in the initial call to the global schema and will not count as transactions tried.

The second measure, transactions completed, indicates how often a transaction finishes, and indirectly how often a transaction will fail. A transaction may fail because communication is too slow, causing an application processor to "time out" when the transaction does not complete within a given time. Or a transaction may fail because the concurrency control algorithm, in order to maintain serializability, aborts the transaction. Both of these situations will be of interest to designers who consider critical data in a combat situation. An incomplete read operation may be considered a serious problem in some scenarios.

Transaction delay, the third collected measure, indicates the distribution of access times to critical and non-critical data. This measure will be of particular interest at nodes which do not store a physical copy of a data record. This is an important issue in determining data availability.

The fourth measure, transaction time in queue, is expected to be largely dependent on the concurrency restrictions placed on processes accessing the same data item at the same physical location. This measure is expected to expose relationships among availability, data consistency, and data currency.

The fifth and sixth measures, DP time by queue and DP processing time, are internal measures which will highlight bottlenecks and provide clues to performance improvement.

ratio of transactions completed to transactions tried is simply a access rate. This ratio will be computed separately for separate types of transactions.

The ratio of messages sent to transactions tried will indicate how a given set of data management algorithms load the communications system. This measure will be most useful when the DBMS is the sole user of the communications network or when it is compared to the same ratio for algorithms with similar non-DBMS message loads.

The last-listed measure of effectiveness, partition detect time, will indicate how effective a schema is in detecting a break in the communications network. This measure is not yet supported in the testbed. Neither is the planned ability to schedule communications partitions as a SIMSCRIPT event.

SUMMARY

The field of distributed database design is still young. It is apparent that the nature of the communications link between remote locations is an important consideration in the design of a distributed DBMS. It is expected that some "tuning" of the data management algorithms to the communications sytem will be required.

The Naval Ocean Systems Center has anticipated the need for a design tool for distributed DBMS built upon a flexible communications network. This

design tool models simple database operations within an existing communications testbed (NASTEE). Flexibility in the communications network is one of the major advantages of this design effort. The standard DBMS architecture is inadequate for the distributed database problem. Honeywell has developed a distributed architecture of DBMS. This architecture provides the basis for the DBMS extensions to NASTEE.

The logical view of a DBMS employs the concept of transactions. Transactions are decomposed into operations on a database by an application processor. The application processor directs these operations to local storage sites. The database is updated or queried by a local data processor. This transaction model has been simulated in SIMSCRIPT II.5. The complexity of transactions has been kept minimal while the basic framework of the simulation is under development. An attempt has also been made to anticipate some of the parameters that are of interest in determining the effectiveness of data management algorithms.

Some of the issues that have yet to be addressed include model validation and future direction of the simulation effort. To address these issues requires more time and resources than were available to this project. Model validation may be attempted by simulating appropriate sections of one of the existing commercial distributed database management systems. Future direction of the simulation will be motivated by the needs of new users.

CONCLUSIONS

The successful design of a distributed DBMS must account for the communications system which will link physically separate locations. This situation may be viewed in two ways: the distributed DMBS may be designed to account for the bandwidth constraint imposed by the communications systems, or the communications system may be designed to accommodate the anticipated message load imposed by the DBMS. For this reason, the extensions to NASTEE described in this report have already attracted interest among both DBMS and communications system designers.

Simulations of the type described in this report require significant amounts of time to write, debug, validate, and run. For this reason, there is a trade-off between simple and complex simulations. Simple simulations save time now; complex simulations may allow more detailed designs and evaluations and save time and money later. "Switches," used in NASTEE to turn program features on or off, appear to be an efficient way of assisting the user to resolve issues of complexity.

RECOMMENDATIONS

In order to make NASTEE DBMS features viable, two steps should be taken: an appropriate section of a commercially available DMBS should be simulated in order to validate the system; a Navy DBMS design should be modeled in order to verify that the features which NASTEE now supports are relevant to Navy systems.

REFERENCES

1. Vineberg, M., Norvell, S., Keune, C., 'Network Architecture Simulated Test and Evaluation Environment," presented at the MIT/ONR Workshop on C^3 Systems, San Diego, Calif., June 1984.

2. "The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Data Base Management Systems," Tsichritzis, D. and Klug, A. eds., AFIPS Press, 210 Summit Ave., Montvale, NJ 07645, 1977.

3. Date, C. J., "An Introduction to Database Systems," Third Edition, Addison-Wesley Publishing Company, Reading, Mass., 1981.

4. Devor, C., Elmari, R., Rahimi, S., "The Design of DDTS: A Testbed For Reliable Distributed Database Management," IEEE 1982 Reliability in Distributed Software and Database Systems, IEE Computer Society Press. Silver Spring, Md., July 1982.

5. Devor, C., Elmari, R., Rahimi, S., et al., "Five-schema Architecture Extends DBMS To Distributed Applications," Electronic Design, March 18 1982.

6. Russel, E., "Building Simulation Models with SIMSCRIPT II.5," CACI, Inc.-Federal. Modeling and Simulation Department. Los Angeles, Calif., January 1983.

END

FILMED

6-85

DTIC